Journal of Aerospace Information Systems, Volume 18, Issue 8, August 2021, pp. 583-
DOI:10.2514/1.I010948

A Fast-Time Demand-Capacity Balancing Optimiser for Collaborative Air Traffic Flow Management

Yan Xu*
Cranfield University, Cranfield MK43 0AL, Bedford, United Kingdom

Leonardo Camargo†, Xavier Prats‡
Technical University of Catalonia, Castelldefels 08860, Barcelona, Spain

I. Introduction

Under current air traffic management (ATM) operations, one of the primary causes for flight delays and congestion is that the number of flights (demand) often exceeds the supply of the airspace accommodation (capacity). The effort thereby to achieve demand and capacity balancing (DCB) is typically known as air traffic flow management (ATFM), which is regarded as an enabler of ATM efficiency and effectiveness [1]. Since the last few decades, a number of researchers have focused their activity on the development of network ATFM models to minimise the congestion costs incurred from demand and capacity imbalances in both airports and airspace volumes (sectors). However, it is important to highlight that the existing network ATFM models present significant challenges in computational tractability when dealing with large-scale problems. As pointed out in [2], this could be one of the main reasons, besides the fairness concern, why they have not been successfully transitioned into practice.

In response to this issue, the computational challenges have been targeted by researchers through applying different numerical computation methods. For the models based on the well-studied Bertsimas Stock-Patterson model [3], a Dantzig-Wolfe decomposition method was proved highly efficient to solve the block-angular formulation [4, 5], where the network traffic flow can be decomposed flight by flight. As reported in [6], the effects of this parallel speed-up were further improved by an implementation of the Graphics Processing Units (GPUs), due to the large amount of cores over the Central Processing Units (CPUs). A similar column generation approach was also adopted in [7], where ground holds, airborne delays, reroutes and cancellations are considered as possible control actions, and the approach was then extended to generate recourse strategies in the presence of uncertain capacity constraints.

With regards to the network ATFM models that are based on Eulerian formulation, a dual-decomposition method was used, which decomposes the traffic flow path by path, and each flight path was solved (optimised) independently [8]. The dimension of a large-scale cell transmission model therefore depends only on the number of identified flight

*Lecturer in ATM/CNS, School of Aerospace, Transport and Manufacturing, Building 320, College Road, Cranfield, Bedfordshire, United Kingdom, Email address: yanxu@cranfield.ac.uk

†PhD Researcher, Department of Physics-Aerospace Division, Office C3-121, Esteve Terradas, 5, Barcelona, Catalonia, Spain, Email address: leonardo@ubihpc.com

‡Serra i Húnter Fellow (Associate Professor), Department of Physics-Aerospace Division, Office C3-104, Esteve Terradas, 5, Barcelona, Catalonia, Spain, Email address: xavier.prats@upc.edu

paths, instead of the total amount of flights. The solution space structure of the problem was explored in [9], proving that an optimal integral solution exists in the LP relaxation that is also the optimal for the original integer program. Considering the nationwide traffic flow management as a link transmission model, a parallel computing framework was proposed, where the model is decomposed into a batch of smaller subproblems that are independently solved on multiple clients coordinated by a server [10]. Then, a big data processing model named Hadoop MapReduce was used in [11] to spread computationally intensive tasks to Hadoop clusters for concurrent executions. Further, a novel layered aggregate model was developed in [12] for handling flexible rerouting problem, which was not well handled in the previous link transmission model. The model can be efficiently computed by a fast cluster architecture as provided by Spark, reported twice as fast as the Hadoop MapReduce model.

In parallel with these efforts devoted to improving the computational efficiency, the ATM community has been also tackling the concurrent evolution of network ATFM models from the operational side. A key concept is the trajectory based operations (TBO), expected to bridge airspace users' (AUs) preferred trajectories and the agreed trajectories resulted from all concerned stakeholders. As one current effort towards incorporating TBO into ATFM practices, the Collaborative Trajectory Options Program (CTOP) developed by the FAA (and AUs) has completed its testing in the National Airspace System [13]. For the existing version of CTOP, a ration-by-schedule (RBS) scheme is adopted, namely flights are assigned the best available routes and slots available at the time flight operators submit their preference requests during the planning phase in a sequential manner [14]. An alternative flight scheduling approach aligned with the collaborative decision-making mechanism, using optimisation instead of pure RBS, has been explored, applying a Max-Min fairness rule to maintain the level of equity [15]. Aimed at the airspace structure in Europe, a collaborative framework was proposed, which integrates AUs' trajectory planning (enabling alternative trajectory options and pre-tactical delay preferences) and the DCB activities performed by the Network Manager (NM) [16]. Further involving the role of air navigation service providers (ANSPs), the framework was extended to synchronise the traffic flow optimisation and airspace configuration scheduling at the same time [17].

These latest operational advances, reflected on the model formulation, impose new challenges to their computational performance. This paper builds on top of the Collaborative ATFM (C-ATFM) framework presented in [16] that demonstrated a potential in significant pra-tactical system delay reduction over the current European ATFM practices. In this paper, we aim to raise the technology readiness level (TRL) of the original works, focusing on improving the tractability of its DCB optimiser. To this end, we apply a Dantzig-Wolfe (DW) decomposition approach and use High Performance Computing (HPC) techniques to attempt the solution in a short time. The main contribution of this paper is to mitigate the computational burden faced by [16, 17], which will make all the contributions demonstrated therein more operationally feasible.

II. Original DCB optimiser

We start from a brief review of the collaborative ATFM framework and the original DCB optimiser presented in [16]. The framework architecture consists of four modules, each representing the tasks that are conducted by either the AUs or the NM, as follows:

- Module I: Initial planning of user-preferred trajectories
- Module II: Detection of demand and capacity imbalance
- Module III: Submission of trajectory options and pre-tactical delay preferences
- Module IV: System-wide optimisation to balance demand and capacity

The last module of the framework deals with a system-wide DCB problem. We will recap its mathematical formulation, which provides a starting point for the application of the decomposition. The original model is formulated with mixed integer linear programming. The time-relevant decision variables are defined as follows:

$$x_{f,t}^{k,j} = \begin{cases} 1, & \text{if } k^{\text{th}} \text{ possible trajectory of flight } f \text{ departs at elementary sector } j\text{'s entrance by time } t \\ 0, & \text{otherwise} \end{cases}$$

$$y_{f,t}^{k,j} = \begin{cases} 1, & \text{if } k^{\text{th}} \text{ possible trajectory of flight } f \text{ arrives at elementary sector } j\text{'s entrance by time } t \\ 0, & \text{otherwise} \end{cases}$$

The above decision variables are used to determine the Controlled Time of Arrival (CTA) for each control point along the trajectory. All the control points and associated CTAs are bonded to the k^{th} trajectory, instead of each flight. An additional set of decision variables z_f^k is considered, such that delays will be imposed on each particular flight, rather than on any of its trajectories that may not be selected. Similar to the concept of Trajectory Options Set (TOS) used in CTOP, each trajectory option will be ranked in the order of user preference indicating their willingness to accept one option over another.

$$z_f^k = \begin{cases} 1, & \text{if flight } f \text{ is assigned with its trajectory by the } k^{\text{th}} \text{ prioritised order} \\ 0, & \text{otherwise} \end{cases}$$

Further illustrative diagrams can be found in [16] where we have elaborated the relations between these decision variables and how they were used to model different DCB initiatives. Also included are the concepts of airspace configuration (e.g., collapsed and elementary sectors) and how the traffic demand within the airspace was counted.

The objective function contains three items, and the total deviation with respect to the initially planned ones is regarded as the extra costs to be minimised: $\min (C_{\Delta F} + C_{\Delta R} + C_{\Delta T})$. Concretely, the extra fuel cost $C_{\Delta F}$ plus the extra route charges $C_{\Delta R}$ can be denoted as follows:

$$C_{\Delta F} + C_{\Delta R} = \sum_{f \in \mathcal{F}} \sum_{k \in \mathcal{K}_f} (\alpha d_f^k + \beta e_f^k)(z_f^k - z_f^{k-1}) \tag{1}$$

where d_f^k represents the extra fuel burn for the k^{th} trajectory of flight f (in comparison to its initial trajectory), being α the price of fuel. Likewise, e_f^k means the extra route charges of the k^{th} trajectory of flight f, being β a weighting cost.

The time related costs $C_{\Delta T}$ are further composed of three items including ground delay (D^g), airborne delay (D^a) and delay recovery (D^r), each of which corresponds to a specific unit cost γ^g, γ^a and γ^r. For convenience, we compute the delay recovery based on the equation $D^r = D^g + D^a - D^e$, where D^e represents the arrival delay at the destination airport. Taking all these into account, we have time relevant costs $C_{\Delta T}$:

$$C_{\Delta T} = \sum_{f \in \mathcal{F}} \sum_{k \in \mathcal{K}_f} [\gamma^g D_k^g + \gamma^a D_k^a - \gamma^r D_k^r] = \sum_{f \in \mathcal{F}} \sum_{k \in \mathcal{K}_f} [(\gamma^g - \gamma^r) D_k^g + (\gamma^a - \gamma^r) D_k^a + \gamma^r D_k^e] \tag{2}$$

By replacing the expressions of D_k^g, D_k^a, D_k^e for each trajectory k, the objective function can be organised in Eq. (3), and the complete formulation of the DCB problem is given as:

$$\begin{aligned} \min \sum_{f \in \mathcal{F}} \sum_{k \in \mathcal{K}_f} \Bigg\{ & (\alpha d_f^k + \beta e_f^k)(z_f^k - z_f^{k-1}) + \sum_{t \in \mathcal{T}_k^{\mathcal{J}_k^{(1)}}} (\gamma^g - \gamma^r)(t - r_f^{k,\mathcal{J}_k^{(1)}})(y_{f,t}^{k,\mathcal{J}_k^{(1)}} - y_{f,t-1}^{k,\mathcal{J}_k^{(1)}}) + \\ & \sum_{j=\mathcal{J}_k^{(m)}, m \in (1,n)} \sum_{t \in \mathcal{T}_k^j} (\gamma^a - \gamma^r) t \left[(x_{f,t}^{k,j} - x_{f,t-1}^{k,j}) - (y_{f,t}^{k,j} - y_{f,t-1}^{k,j}) \right] + \sum_{t \in \mathcal{T}_k^{\mathcal{J}_k^{(n)}}} \gamma^r (t - r_f^{k,\mathcal{J}_k^{(n)}})^{(1+\epsilon)} (x_{f,t}^{k,\mathcal{J}_k^{(n)}} - x_{f,t-1}^{k,\mathcal{J}_k^{(n)}}) \Bigg\}, \end{aligned} \tag{3}$$

$$\text{s.t.} \quad z_f^{\underline{\mathcal{K}}_f - 1} = 0, \quad z_f^{\overline{\mathcal{K}}_f} = 1 \quad \forall f \in \mathcal{F}, \tag{4}$$

$$z_f^k - z_f^{k-1} \geq 0, \quad \forall f \in \mathcal{F}, \forall k \in \mathcal{K}_f, \tag{5}$$

$$x_{f,\underline{\mathcal{T}}_k^j - 1}^{k,j} = y_{f,\underline{\mathcal{T}}_k^j - 1}^{k,j} = 0, \quad x_{f,\overline{\mathcal{T}}_k^j}^{k,j} = y_{f,\overline{\mathcal{T}}_k^j}^{k,j} = z_f^k - z_f^{k-1} \quad \forall f \in \mathcal{F}, \forall k \in \mathcal{K}_f, \forall j \in \mathcal{J}_k, \tag{6}$$

$$x_{f,t}^{k,j} - x_{f,t-1}^{k,j} \geq 0, \quad y_{f,t}^{k,j} - y_{f,t-1}^{k,j} \geq 0, \quad x_{f,t}^{k,j} - y_{f,t}^{k,j} \leq 0 \quad \forall f \in \mathcal{F}, \forall k \in \mathcal{K}_f, \forall j \in \mathcal{J}_k, \forall t \in \mathcal{T}_k^j, \tag{7}$$

$$y^{k,j'}_{f,t+\underline{T}^{jj'}_k} - x^{k,j}_{f,t} \leq 0 \quad \forall f \in \mathcal{F}, \forall k \in \mathcal{K}_f, j = \mathcal{J}^{(m)}_k, j' = \mathcal{J}^{(m+1)}_k, \forall m \in [1,n), \forall t \in \mathcal{T}^j_k \cap [\underline{\mathcal{T}}^{j'}_k - \underline{T}^{jj'}_k, \overline{\mathcal{T}}^{j'}_k - \underline{T}^{jj'}_k], \quad (8)$$

$$x^{k,j}_{f,t+\overline{T}^{jj'}_k} - y^{k,j}_{f,t} \geq 0 \quad \forall f \in \mathcal{F}, \forall k \in \mathcal{K}_f, j = \mathcal{J}^{(m)}_k, j' = \mathcal{J}^{(m+1)}_k, \forall m \in [1,n), \forall t \in [\underline{\mathcal{T}}^j_k, \overline{\mathcal{T}}^j_k - \overline{T}^{jj'}_k], \quad (9)$$

$$\sum_{f \in \mathcal{F}} \sum_{k \in \mathcal{K}_f} \sum_{t \in \mathcal{T}^{\mathcal{J}^\tau_{k,l}}_k \cap \mathcal{T}(\tau)} (x^{k,\mathcal{J}^\tau_{k,l}}_{f,t} - x^{k,\mathcal{J}^\tau_{k,l}}_{f,t-1}) \leq C^\tau_l \quad \forall \tau \in \mathbb{T}, \forall l \in \mathcal{L}_\tau, \quad (10)$$

$$x^{k,j}_{f,t},\ y^{k,j}_{f,t} \in \{0,1\} \quad \forall f \in \mathcal{F}, \forall k \in \mathcal{K}_f, \forall j \in \mathcal{J}_k, \forall t \in \mathcal{T}^j_k, \quad (11)$$

$$z^k_f \in \{0,1\} \quad \forall f \in \mathcal{F}, \forall k \in \mathcal{K}_f. \quad (12)$$

Constraints (4)-(5) enforce that only one trajectory of all submitted options (including the initial and the alternatives) will be selected for each flight. Constraints (6)-(7) guarantee that each selected trajectory k, is assigned with only one time slot for departing and arriving respectively at position j within a prescribed time window $\mathcal{T}^j_k$. It also specifies that the departure time $x^{k,j}_{f,t}$ is not earlier than the arrival time $y^{k,j}_{f,t}$. Constraints (8) and (9) stipulate the time bounds of delay recovery and airborne delay. Constraint (10) ensures that the traffic demand does not exceed the capacity of each airspace sector. The opening scheme of operating sectors $\mathcal{L}_\tau$ must be taken into account in which capacity is a function of time (refer to [16] for more details of how the dynamic capacity is modelled). Finally, Constraints (11) and (12) specify the binary constraints. For more details about this formulation, the reader may refer to [16].

III. DW decomposition

DW decomposition is a classic solution approach for structured LP problems that form the block angular structure [18]. A master problem is devised which only concentrates on the coupling constraints, and the subproblems are solved individually. This suits particularly well with the previously formulated DCB model, where the flight-specific constraints can be treated as subproblems, and the airspace capacity can be regarded as the coupling constraints, which as a whole forms the block angular structure. We will detail how it is implemented to the specific DCB formulation presented in Sec. II.

A. DW decomposition principle

To apply the DW decomposition, the model needs to respect the following block angular (or primal block) structure. For illustrative purpose, the nomenclature used in Sec. III.A will be self-contained.

$$\min \ \boldsymbol{c}_0^T \boldsymbol{x}_0 + \boldsymbol{c}_1^T \boldsymbol{x}_1 + \cdots + \boldsymbol{c}_k^T \boldsymbol{x}_k \tag{13}$$

$$\text{s.t.} \begin{bmatrix} B_0 & B_1 & B_2 & \dots & B_k \\ & A_1 & & & \\ & & A_2 & & \\ & & & \ddots & \\ & & & & A_k \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_0 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_0 \\ \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \vdots \\ \boldsymbol{b}_k \end{bmatrix} \tag{14}$$

$$\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_k \geq 0 \tag{15}$$

where $\{\boldsymbol{x}_k |\ k \in [0, 1, \dots, K]\}$ are the decision variables. $\boldsymbol{c}_k^T$ represents the vector of weighted cost for $\boldsymbol{x}_k$, while B_k, A_k and $\boldsymbol{b}_k$ are the associated constant vectors of $\boldsymbol{x}_k$ in constraints. The constraints $\sum_{k=0}^{n} B_k \boldsymbol{x}_k = \boldsymbol{b}_0$ corresponding to the top row of sub-matrices are called the coupling constraints.

Consider the feasible region $P = \{\boldsymbol{x} |\ A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0\}$ and assume P is bounded (as is the case for the DCB model where all variables are bounded in $\{0, 1\}$). According to the Minkowski's Representation Theorem [19], any point $\boldsymbol{x} \in P$ can be denoted by a linear combination of its extreme points $\boldsymbol{x}^{(i)}$ as follows: $\boldsymbol{x} = \sum_i \lambda_i \boldsymbol{x}^{(i)}$, and the sum of all λ_i should be equal to 1, namely $\sum_i \lambda_i = 1, \lambda_i \geq 0$, which is also known as the convexity constraint. In this case, we can formulate the problem in terms of variables λ_i instead of $\boldsymbol{x}$.

Then, the problem can be decoupled into a set of subproblems (for $\{\boldsymbol{x}_k |\ k \in [1, \dots, K]\}$) and a master problem. The subproblems deal with the constraints: $A_k \boldsymbol{x}_k = \boldsymbol{b}_k,\ \boldsymbol{x}_k \geq 0$, whilst the master problem contains the following equations: $\min \sum_k \boldsymbol{c}_k^T \boldsymbol{x}_k$, s.t. $\sum_k B_k \boldsymbol{x}_k = \boldsymbol{b}_0,\ \boldsymbol{x}_0 \geq 0$. Next, we can apply the above convexity constraint to the subproblems (let I_k be the set of extreme points for subproblem k), and substitute them into the master problem, which will generate:

$$\min \boldsymbol{c}_0^T \boldsymbol{x}_0 + \sum_{k=1}^{K} \sum_{i=1}^{I_k} (\boldsymbol{c}_k^T \boldsymbol{x}_k^{(i)}) \lambda_{k,i}, \tag{16}$$

$$B_0 \boldsymbol{x}_0 + \sum_{k=1}^{K} \sum_{i=1}^{I_k} (B_k \boldsymbol{x}_k^{(i)}) \lambda_{k,i} = \boldsymbol{b}_0, \tag{17}$$

$$\sum_{i=1}^{I_k} \lambda_{k,i} = 1, \forall k \in \{1, 2, \ldots, K\} \tag{18}$$

$$\boldsymbol{x}_0,\ \lambda_{k,i} \geq 0 \tag{19}$$

Although the number of rows is reduced, the number of extreme points and rays $\boldsymbol{x}^{(i)}$ of each subproblem is large, resulting in an enormous number of variables $\lambda_{k,i}$. However, most of these variables will be non-basic at zero, and need not be involved in the problem, so the idea is that only variables with a promising reduced cost will be considered, being the others fixed to zero, which in turn forms a restricted master problem. This problem will not be fixed in size, and new variables (with promising reduced cost) will be added into it within each loop of executing the so-called delayed column generation algorithm, as shown in Fig. 1.

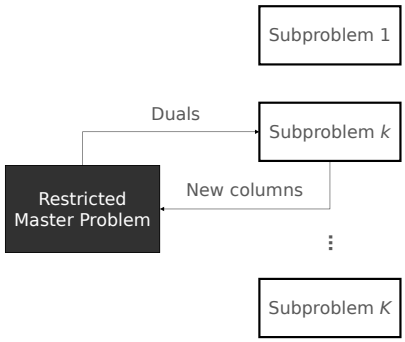


Fig. 1 Delayed column generation algorithm.

To measure if a variable $\lambda_{k,i}$ has a promising reduced cost for the restricted master problem, we consider $\boldsymbol{\pi}_1$ as the dual variables for the coupling constraint whilst $\boldsymbol{\pi}_2^k$ as those for the convexity constraints. Then, the reduced cost $\sigma_{k,i}$ can be derived as follows:

$$\sigma_{k,i} = (\boldsymbol{c}_k^T - \boldsymbol{\pi}_1^T B_k) \boldsymbol{x}_k^{(i)} - \boldsymbol{\pi}_2^k \tag{20}$$

Therefore, the most attractive basic feasible solution $\boldsymbol{x}_k$ to enter the restricted master problem is found by maximising the reduced cost of the following LP problem. For each subproblem k, if $\sigma_k < 0$ then we can introduce a new column $\lambda_{k,i}$ to the restricted master problem.

$$\min \sigma_k = (\boldsymbol{c}_k^T - \boldsymbol{\pi}_1^T B_k)\boldsymbol{x}_k - \boldsymbol{\pi}_2^k \tag{21}$$

$$A_k \boldsymbol{x}_k = \boldsymbol{b}_k \tag{22}$$

$$\boldsymbol{x}_k \geq 0 \tag{23}$$

B. Flight-specific subproblem

Following the original Constraints (4) - (9), we can decompose the flight-specific constraints by each flight f and derive the following subproblem formulation. For all the different flights, the solution can be done simultaneously.

$$\hat{s}_1 = -\left[\sum_{\tau\in\mathbb{T}}\sum_{l\in\mathcal{L}_\tau}\sum_{k\in\hat{\mathcal{K}}}\sum_{t\in\mathcal{T}_k^{\mathcal{J}_{k,l}^\tau}\cap\mathcal{T}(\tau)} \pi_1^{\tau,l}(\hat{x}_t^{k,\mathcal{J}_{k,l}^\tau} - \hat{x}_{t-1}^{k,\mathcal{J}_{k,l}^\tau})\right] - \hat{\pi}_2^x - \hat{\pi}_2^y - \hat{\pi}_2^z, \tag{24}$$

$$\begin{aligned}\hat{s}_2 = \sum_{k\in\hat{\mathcal{K}}}\Bigg\{&(\alpha\hat{d}^k + \beta\hat{e}^k)(\hat{z}^k - \hat{z}^{k-1}) + \sum_{t\in\mathcal{T}_k^{\mathcal{J}_k^{(1)}}}(\gamma^g - \gamma^r)(t - \hat{r}^{k,\mathcal{J}_k^{(1)}})(\hat{y}_t^{k,\mathcal{J}_k^{(1)}} - \hat{y}_{t-1}^{k,\mathcal{J}_k^{(1)}}) + \\ &\sum_{j=\mathcal{J}_k^{(m)},m\in(1,n)}\sum_{t\in\mathcal{T}_k^j}(\gamma^a - \gamma^r)t\left[(\hat{x}_t^{k,j} - \hat{x}_{t-1}^{k,j}) - (\hat{y}_t^{k,j} - \hat{y}_{t-1}^{k,j})\right] + \sum_{t\in\mathcal{T}_k^{\mathcal{J}_k^{(n)}}}\gamma^r(t - \hat{r}^{k,\mathcal{J}_k^{(n)}})^{(1+\epsilon)}(\hat{x}_t^{k,\mathcal{J}_k^{(n)}} - \hat{x}_{t-1}^{k,\mathcal{J}_k^{(n)}})\Bigg\} \\ &-\left[\sum_{\tau\in\mathbb{T}}\sum_{l\in\mathcal{L}_\tau}\sum_{k\in\hat{\mathcal{K}}}\sum_{t\in\mathcal{T}_k^{\mathcal{J}_{k,l}^\tau}\cap\mathcal{T}(\tau)} \pi_1^{\tau,l}(\hat{x}_t^{k,\mathcal{J}_{k,l}^\tau} - \hat{x}_{t-1}^{k,\mathcal{J}_{k,l}^\tau})\right] - \hat{\pi}_2^x - \hat{\pi}_2^y - \hat{\pi}_2^z,\end{aligned} \tag{25}$$

$$\text{s.t.} \quad \hat{z}^{\underline{\hat{\mathcal{K}}}-1} = 0, \quad z^{\overline{\hat{\mathcal{K}}}} = 1, \tag{26}$$

$$\hat{z}^k - \hat{z}^{k-1} \geq 0, \quad \forall k \in \hat{\mathcal{K}}, \tag{27}$$

$$\hat{x}^{k,j}_{\underline{\mathcal{T}}_k^j-1} = \hat{y}^{k,j}_{\underline{\mathcal{T}}_k^j-1} = 0, \quad \hat{x}^{k,j}_{\overline{\mathcal{T}}_k^j} = \hat{y}^{k,j}_{\overline{\mathcal{T}}_k^j} = \hat{z}^k - \hat{z}^{k-1} \quad \forall k \in \hat{\mathcal{K}}, \forall j \in \mathcal{J}_k, \tag{28}$$

$$\hat{x}_t^{k,j} - \hat{x}_{t-1}^{k,j} \geq 0, \quad \hat{y}_t^{k,j} - \hat{y}_{t-1}^{k,j} \geq 0, \quad \hat{x}_t^{k,j} - \hat{y}_t^{k,j} \leq 0 \quad \forall k \in \hat{\mathcal{K}}, \forall j \in \mathcal{J}_k, \forall t \in \mathcal{T}_k^j, \tag{29}$$

$$\hat{y}^{k,j'}_{t+\underline{T}_k^{jj'}} - \hat{x}^{k,j}_t \leq 0 \quad \forall k \in \hat{\mathcal{K}}, j = \mathcal{J}_k^{(m)}, j' = \mathcal{J}_k^{(m+1)}, \forall m \in [1,n), \forall t \in \mathcal{T}_k^j \cap [\underline{\mathcal{T}}_k^{j'} - \underline{T}_k^{jj'}, \overline{\mathcal{T}}_k^{j'} - \underline{T}_k^{jj'}], \tag{30}$$

$$\hat{x}^{k,j}_{t+\overline{T}_k^{jj'}} - \hat{y}^{k,j}_t \geq 0 \quad \forall k \in \hat{\mathcal{K}}, j = \mathcal{J}_k^{(m)}, j' = \mathcal{J}_k^{(m+1)}, \forall m \in [1,n), \forall t \in [\underline{\mathcal{T}}_k^j, \overline{\mathcal{T}}_k^j - \overline{T}_k^{jj'}], \tag{31}$$

$$\hat{x}^{k,j}_t,\ \hat{y}^{k,j}_t \geq 0 \quad \forall k \in \hat{\mathcal{K}}, \forall j \in \mathcal{J}_k, \forall t \in \mathcal{T}_k^j, \tag{32}$$

$$\hat{z}^k \geq 0 \quad \forall k \in \hat{\mathcal{K}}. \tag{33}$$

Eqs. (24) - (25) present two objective functions, corresponding to Phase 1 and Phase 2 of the solution algorithm. The dual variable of the coupling constraints is depicted as $\pi_1^{\tau,l}$, and the duals of the convexity constraints (for the new decision variables $\lambda^x_{f,i}$, $\lambda^y_{f,i}$ and $\lambda^z_{f,i}$ of the master problem) are denoted by $\hat{\pi}^x_2$, $\hat{\pi}^y_2$ and $\hat{\pi}^z_2$ respectively. Constraints (26) - (31) are in line with those of the original problem. The only difference is that all the constraints here are irrelevant to any flight f. Constraints (32) - (33) state that the decision variables of the subproblems are non-negative real numbers. Given Constraints (26) - (29), the actual region will be bounded to $[0, 1]$.

C. Restricted master problem

The restricted master problem is formulated below. In order to realise the effects of delayed column generation, we define a new set of tentative proposals $i \in \mathcal{I}$ and its subset $\Theta \subset \mathcal{I}$ representing the valid proposals for variables or weights $\lambda^x_{f,i}$, $\lambda^y_{f,i}$ and $\lambda^z_{f,i}$, as long as they have some promising reduced cost. This subset of proposals Θ_f for each subproblem is a dynamic set, as additional valid proposals might be further included in each loop.

$$m_1 = \mu, \tag{34}$$

$$m_2 = \sum_{f \in \mathcal{F}} \sum_{i \in \Theta_f} \left(c^x_{f,i} \lambda^x_{f,i} + c^y_{f,i} \lambda^y_{f,i} + c^z_{f,i} \lambda^z_{f,i} \right), \tag{35}$$

$$\sum_{f \in \mathcal{F}} \sum_{i \in \Theta_f} B^{\tau,l}_{f,i}\ \lambda^x_{f,i} \leq C^\tau_l + \mu \quad \forall \tau \in \mathbb{T}, \forall l \in \mathcal{L}_\tau, \tag{36}$$

$$\sum_{i\in\Theta_f} \lambda^x_{f,i} = 1, \quad \sum_{i\in\Theta_f} \lambda^y_{f,i} = 1, \quad \sum_{i\in\Theta_f} \lambda^z_{f,i} = 1 \quad \forall f \in \mathcal{F}, \tag{37}$$

$$\lambda^x_{f,i} = \lambda^y_{f,i} = \lambda^z_{f,i} \geq 0 \quad \forall f \in \mathcal{F}, \forall i \in \Theta_f. \tag{38}$$

Eqs. (34) - (35) present two objective functions, corresponding to Phase 1 and Phase 2 of the algorithm as well. The objective function of the original problem considers three sets of variables $x^{k,j}_{f,t}$, $y^{k,j}_{f,t}$ and z^k_f but the coupling (capacity) constraint concerns only $x^{k,j}_{f,t}$.

$$B^{\tau,l}_{f,i} = \sum_{k\in\mathcal{K}_f} \sum_{t\in\mathcal{T}^{\mathcal{T}^\tau_{k,l}}_k \cap \mathcal{T}(\tau)} (\mathrm{x}^{k,\mathcal{T}^\tau_{k,l}}_{f,t} - \mathrm{x}^{k,\mathcal{T}^\tau_{k,l}}_{f,t-1}) \quad \forall f \in \mathcal{F}, \forall i \in \Theta_f, \forall \tau \in \mathbb{T}, \forall l \in \mathcal{L}_\tau, \tag{39}$$

$$\begin{aligned} c^x_{f,i} = &\sum_{k\in\mathcal{K}_f} \sum_{t\in\mathcal{T}^{\mathcal{J}^{(n)}_k}_k} \gamma^r (t - r^{k,\mathcal{J}^{(n)}_k}_f)^{(1+\epsilon)} (\mathrm{x}^{k,\mathcal{J}^{(n)}_k}_{f,t} - \mathrm{x}^{k,\mathcal{J}^{(n)}_k}_{f,t-1}) + \\ &\sum_{k\in\mathcal{K}_f} \sum_{j=\mathcal{J}^{(m)}_k, m\in(1,n)} \sum_{t\in\mathcal{T}^j_k} (\gamma^a - \gamma^r) t (\mathrm{x}^{k,j}_{f,t} - \mathrm{x}^{k,j}_{f,t-1}) \quad \forall f \in \mathcal{F}, \forall i \in \Theta_f, \end{aligned} \tag{40}$$

$$\begin{aligned} c^y_{f,i} = &\sum_{k\in\mathcal{K}_f} \sum_{t\in\mathcal{T}^{\mathcal{J}^{(1)}_k}_k} (\gamma^g - \gamma^r)(t - r^{k,\mathcal{J}^{(1)}_k}_f)(\mathrm{y}^{k,\mathcal{J}^{(1)}_k}_{f,t} - \mathrm{y}^{k,\mathcal{J}^{(1)}_k}_{f,t-1}) - \\ &\sum_{k\in\mathcal{K}_f} \sum_{j=\mathcal{J}^{(m)}_k, m\in(1,n)} \sum_{t\in\mathcal{T}^j_k} (\gamma^a - \gamma^r) t (\mathrm{y}^{k,j}_{f,t} - \mathrm{y}^{k,j}_{f,t-1}) \quad \forall f \in \mathcal{F}, \forall i \in \Theta_f, \end{aligned} \tag{41}$$

$$c^z_{f,i} = \sum_{k\in\mathcal{K}_f} (\alpha d^k_f + \beta e^k_f)(\mathrm{z}^k_f - \mathrm{z}^{k-1}_f) \quad \forall f \in \mathcal{F}, \forall i \in \Theta_f, \tag{42}$$

$$\xi^{x,k,j}_{i,f,t} = \mathrm{x}^{k,j}_{f,t} - \mathrm{x}^{k,j}_{f,t-1} \quad \forall f \in \mathcal{F}, \forall i \in \Theta_f, \forall k \in \mathcal{K}_f, \forall j \in \mathcal{J}_k, \forall t \in \mathcal{T}^j_k, \tag{43}$$

$$\xi^{y,k,j}_{i,f,t} = \mathrm{y}^{k,j}_{f,t} - \mathrm{y}^{k,j}_{f,t-1} \quad \forall f \in \mathcal{F}, \forall i \in \Theta_f, \forall k \in \mathcal{K}_f, \forall j \in \mathcal{J}_k, \forall t \in \mathcal{T}^j_k, \tag{44}$$

$$\xi^{z,k}_{i,f} = \mathrm{z}^k_f - \mathrm{z}^{k-1}_f \quad \forall f \in \mathcal{F}, \forall i \in \Theta_f, \forall k \in \mathcal{K}_f, \tag{45}$$

Three proposed costs $c^x_{f,i}$, $c^y_{f,i}$ and $c^z_{f,i}$ are in the objective function Eq. (35). Their calculations, using the solutions of $\mathrm{x}^{k,j}_{f,t}$, $\mathrm{y}^{k,j}_{f,t}$ and z^k_f that are derived from solving the subproblems, are given in Eqs. (40) - (42). The coupling Constraint (36) requires only one coefficient for $\lambda^x_{f,i}$, namely $B^{\tau,l}_{f,i}$ and its expression can be found in Eq. (39). In addition,

Constraint (37) specifies the convexity constraints for $\lambda^x_{f,i}$, $\lambda^y_{f,i}$ and $\lambda^z_{f,i}$ respectively. Constraint (38) synchronises the solutions for the three sets of decision variables appearing in the same subproblem f and sharing the same proposal i, and also states that they are all subject to non-negative real numbers. Finally, through each iteration of the algorithm, we also record the solutions given by any feasible proposals, as shown in Eqs. (43) - (45). The recorded data will be useful when recovering the final solution for the original problem, as discussed in Sec. III.E.

Algorithm 1 DW-DCB pseudo algorithm.

{*Initialisation*}
Set $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2^f = 0$ and choose initial proposals Θ_f
{*Phase 1*}
while true **do**
    Solve the restricted master problem minimising m_1
    $\boldsymbol{\pi}_1 \leftarrow$ duals of coupling constraints
    $\boldsymbol{\pi}_2^f \leftarrow$ duals of f^{th} convexity constraint
    if $m_1 \rightarrow 0$ **then**
        Fix μ to 0
        Break: switch to Phase 2
    else
        for $f = 1, 2, \ldots, F$ **do**
            Plug $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2^f$ into f^{th} subproblem
            Solve each subproblem minimising s_1^f
            if $s_1^f < 0$ **then**
                Add proposal i to Θ_f
{*Phase 2*}
while true **do**
    Solve the restricted master problem minimising m_2
    $\boldsymbol{\pi}_1 \leftarrow$ duals of coupling constraints
    $\boldsymbol{\pi}_2^f \leftarrow$ duals of f^{th} convexity constraint
    for $f = 1, 2, \ldots, F$ **do**
        Plug $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2^f$ into f^{th} subproblem
        Solve each subproblem minimising s_2^f
        if $s_2^f < 0$ **then**
            Add proposal i to Θ_f
    if no new proposals generated **then**
        Break: optimal solution found

D. Pseudo algorithm

The DW-DCB pseudo code is presented in Algorithm 1, which is divided into three main phases: Initialisation, Phase 1 and Phase 2. In terms of Initialisation, the subproblems will be first computed for once, where the duals $\boldsymbol{\pi}_1, \boldsymbol{\pi}_2^f$ are set to 0, and all the solutions will be used to generate the initial set of valid proposals Θ_f subject to iterations. Note that if any of the subproblems is infeasible, the original problem proves infeasible.

Next, as the initial proposals may violate the coupling Constraint (36), we introduce a non-negative artificial variable μ to the right-hand side and minimise it during Phase 1, as depicted by objective function m_1 in Eq. (34). Accordingly, the reduced cost for the subproblems in Phase 1, as given by objective function $\hat{s}_1$ in Eq. (24), does not concern the

coefficient of the coupling constraint with $\boldsymbol{c}_k^T = 0$). After a certain amount of iterations when m_1 is approximately equal to 0, we can switch to Phase 2 in which μ needs to be removed (or fixed to 0). Otherwise, if m_1 still remains greater than 0 after certain iterations, the original problem may end up infeasible as well.

In phase 2, given the last solutions of subproblems and current valid proposals, the restricted master problem will be recomputed minimising m_2 hereafter. Likewise, the objective of the subproblems will change to $\hat{s}_2$, where a complete coefficient of the coupling constraint must be considered, as presented in Eq. (35). The iteration should keep looping whilst new proposals are added if their corresponding reduced costs are negative (i.e., $s_2^f < 0$). We accept all columns with potential to improve the master problem's objective, but other options may choose the proposals with only the greatest reduced costs. In any case, the algorithm will eventually stop in a condition that no new proposals are generated, namely all subproblems' objectives are non-negative, which means the optimal solution is found.

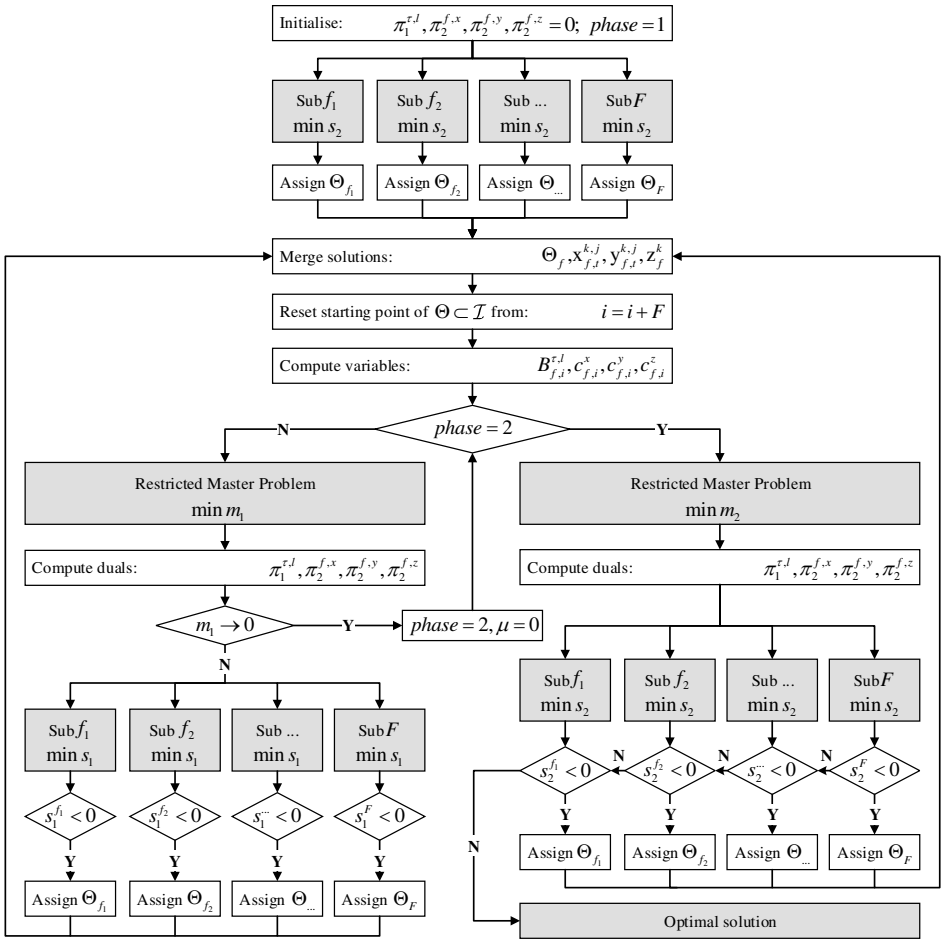


Fig. 2 Schematic of the DW-DCB algorithm.

In light of the pseudo code presented in Algorithm 1, we have included a more detailed diagram in Fig. 2 to demonstrate the whole work flow (using the same nomenclature in problem formulation) and to illustrate how the exact

input/output are exchanged between each subporblem and the master problem. The key that drives the algorithm's iterations is the valid proposal set Θ_f. To have a clear track of each subproblem f and its mapping proposal, we reset the starting point of Θ after each loop within the general proposal set $\mathcal{I}$.

E. Solution recovery

Once we have found the optimal solution using the DW-DCB algorithm, next is to recover the final results with regards to the original MILP model. The original variables $x_{f,t}^{k,j}, y_{f,t}^{k,j}, z_f^k$ can be computed as follows, where $\xi_{i,f,t}^{x,k,j}, \xi_{i,f,t}^{y,k,j}, \xi_{i,f}^{z,k}$ are recorded by Eqs. (43) - (45).

$$x_{f,t}^{k,j} - x_{f,t-1}^{k,j} = \sum_{i \in \Theta_f} \lambda_{f,i}^{x} \xi_{i,f,t}^{x,k,j} \quad \forall f \in \mathcal{F}, \forall k \in \mathcal{K}_f, \forall j \in \mathcal{J}_k, \forall t \in \mathcal{T}_k^j, \tag{46}$$

$$y_{f,t}^{k,j} - y_{f,t-1}^{k,j} = \sum_{i \in \Theta_f} \lambda_{f,i}^{y} \xi_{i,f,t}^{y,k,j} \quad \forall f \in \mathcal{F}, \forall k \in \mathcal{K}_f, \forall j \in \mathcal{J}_k, \forall t \in \mathcal{T}_k^j, \tag{47}$$

$$z_f^k - z_f^{k-1} = \sum_{i \in \Theta_f} \lambda_{f,i}^{z} \xi_{i,f}^{z,k} \quad \forall f \in \mathcal{F}, \forall k \in \mathcal{K}_f. \tag{48}$$

Given that the values for all the boundary variables (namely $x_{f,\underline{\mathcal{T}}_k^j-1}^{k,j}$, $y_{f,\underline{\mathcal{T}}_k^j-1}^{k,j}$, $z_f^{\underline{\mathcal{K}}_f-1}$) are fixed to 0, the above recursive Eqs. (46) - (48) can be easily solved, thus generating the solution for each individual variable. However, this solution is not necessarily subject to the integral (binary) constraint. In order to maintain the integrality of the original problem, there are different options (see [20] for instance). In this paper, we only test an easy yet highly efficient method (similar as done in [5]), which is to round them to the nearest integer.

We first deal with the set of trajectory-relevant variables z_f^k only, applying directly the round to nearest method (i.e., 0 or 1 in this case). Since the variables always satisfy Constraints (4) and (5), it will not break such constraints even after rounding each variable to an integer. Based on that, we move on to tackle the time-relevant variables $x_{f,t}^{k,j}, y_{f,t}^{k,j}$. For every possible (f,k) in terms of $(z_f^k - z_f^{k-1})$ that represents if trajectory k is selected for flight f, we will check its integrality. If it equals to 0 or 1 already, then we directly round the associated time-relevant variables. Otherwise, we first re-scale them based on the solved $(z_f^k - z_f^{k-1})$ and its rounded value $(Z_f^k - Z_f^{k-1})$, and then apply rounding. This is because the upper bound of time-relevant variables are determined by the value of trajectory variables (see Constraint (6)), which in turn should be re-defined by the rounded value (either 0 or 1).

Moreover, the reason why the rounding could be efficient is that all the flight-specific Constraints (4) - (9) will be always feasible after the rounding process, which is important. The rounded assignment to variables will maintain the satisfaction of these physical/temporal constraints. The only condition that might be broken is the capacity Constraint (10). Violating such constraint to a small extent, however, does not necessarily affect the solution's feasibility in reality.

In fact, it has been observed that a certain amount of capacity overloads are usually allowed (and in some cases the allowance can be large). As explained in [21], this could be due to various operational reasons, such as the lack of initial schedules for pop-up flights, the conservative method for capacity evaluation, and the current way of counting traffic demand (i.e., flight entry rate) without considering the factors of occupancy, traffic pattern and complexity. In Sec. V.B.2, a detailed assessment of this violation is given through a group of real-world case studies.

IV. HPC deployment

We consider three levels of HPC parallelism, involving the coarse-grained and the fine-grained, as well as a medium-grained level, as shown in Fig. 3. Specifically, the subproblems are first divided into a set of groups (i.e., Group 0, 1, 2, ..., X), based on the total number of available processors of a cluster of computers, namely the coarse-grained level. On top of each subproblem group, we then apply a grid computing facility at the medium-grained level to avoid serially looping over all subproblems of that group. At the fine-grained level, we adopt a multi-threading LP solver to tackle the growing complex master problem.

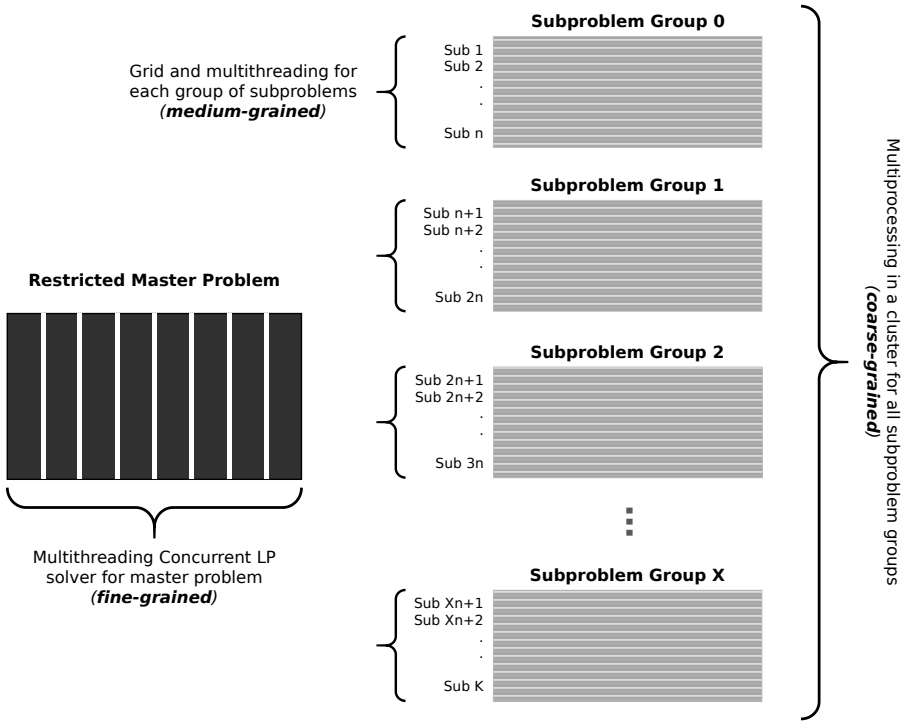


Fig. 3 Three levels of HPC parallelism implemented for the DW-DCB algorithm.

A. Multi-processing

To execute the DW-DCB algorithm among a cluster of computers, as per the coarse-grained parallelism, we developed a DW-DCB-HPC program taking advantage of MPI (Message Passing Interface) software libraries, specifically the

Python library mpi4py [22], which is able to use any quantity of parallel computing processes. Equivalently to an HPC cluster, where nodes are classified as master and slaves, the program presented here can make use of a parallel master process and a set of parallel slave processes. The master process is in charge of collecting and merging the slave processes results.

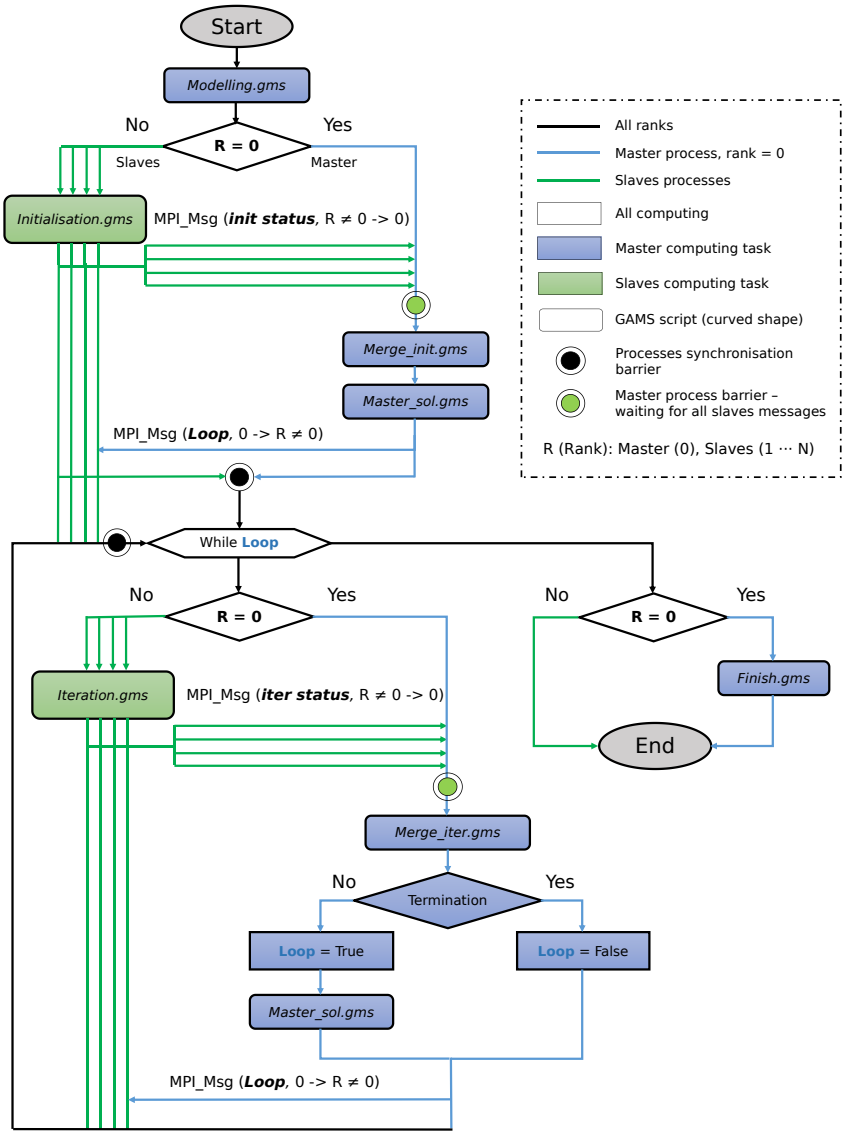


Fig. 4 Workflow of the in-house DW-DCB-HPC program.

This program has two subsets of processes. The first subset consists of the master process, while the second to the slave processes with ranks 1 to N, having N parallel processes. All processes perform computation over a different data subset, such as those belonging to different subproblem groups. The general workflow of the program is as presented in Fig. 4. All processes start the execution of the program after being assigned a rank, from 0 to N, having N parallel

processes. Depending on their specific rank, processes will proceed with its corresponding computing tasks. All the tasks and interactions are coded in GAMS [23] (in terms of *.gms scripts), within which GUROBI [24] is called wherever needed as the LP solver.

Specifically, the master process starts with the *Modelling* script to generate the generic models over all subproblems and the restricted master problem, using the input data introduced in Sec. V.A, generating a GAMS pre-complied (g00) file. Then, the slave processes execute simultaneously the *Initialisation* script, where each process performs it on a different data subset. Once all the slaves are finished, they inform the master process with their corresponding MPI_Msg (init status). As soon as all individual status are received at the master process, two computing tasks are executed, namely *Merge_init* and *Master_sol*. Following this Initialisation phase (see Algorithm 1), all processes enter a loop cycle that is for Phase 1 and Phase 2, with similar tasks depending on specific ranks, until a termination condition is satisfied (before which the switch condition from Phase 1 to Phase 2 is embedded in *Master_sol*). Afterwards, the master process recovers the final results with the *Finish* script.

B. Grid computing and multi-threading

According to [23], when GAMS encounters a solve statement during execution it proceeds in three basic steps:

- **Generation**: The symbolic equations are used to instantiate the model using the current state of the data base (as done with the *Modelling* script), which contains all needed by a solution method to attempt a solution;
- **Solution**: The generated model instance (i.e., each subproblem in this case) is handed over to a solver and GAMS will wait until it terminates; and
- **Update**: The detailed solution and statistics (for each solved subproblem) are passed back to GAMS from the solver to update the GAMS data base.

Most of the *generation* tasks in the DW-DCB-HPC program can be done prior to the main computing work. Every time some parts of the model have to be updated in the loop (recall Algorithm 1), we only need to re-generate a few more symbolic equations to overwrite the initial ones, which does not require a notable amount of time. The other GAMS *update* process is fast as well, taking into account that only a small portion of the data base need to be updated and then passed to the next round of iteration. In other words, when solving our problem, the time required for *generation* and *update* will be much smaller than the *solution* time. Given that the subproblems do not depend on each other, we can solve them in parallel and update the data base in a random order. Therefore, what we need is to re-generate a few more equations for modelling, submit them for solution and continue. At a certain point, we will collect the completed solutions and update the data base accordingly. To this end, the following two loops are included:

- **Submission loop**: In this loop we conduct model *generation* (for all the subproblems of each group) and submit them for *solution* that can be executed independently; and
- **Collection loop**: After the above loop is finished, the solutions of the previously submitted models are collected

as soon as a solution is available for the *update* of data base.

During the above loop, the multi-threading facility can be activated (as the *solution* process can be done independently) for asynchronous solves, where a separate thread will be used for the solver to handle the submitted model instance, whilst allowing efficient in-memory communication with GAMS.

The last level of the fine-grained parallelism is reflected on solving the restricted master problem, which becomes more and more complex as new columns are introduced over the iterations. This can be easily realised by setting up the multiple threads function for the solver to apply a concurrent solve. It will run different optimisation algorithms (such as primal and dual simplex and barrier) on different threads, and return when the first one finishes [24]. Note that the same way might be adopted to solve the subproblems, but not necessarily efficient because each subproblem is easy to solve in our problem and thus can be handled in a straightforward way.

V. Experiments and results

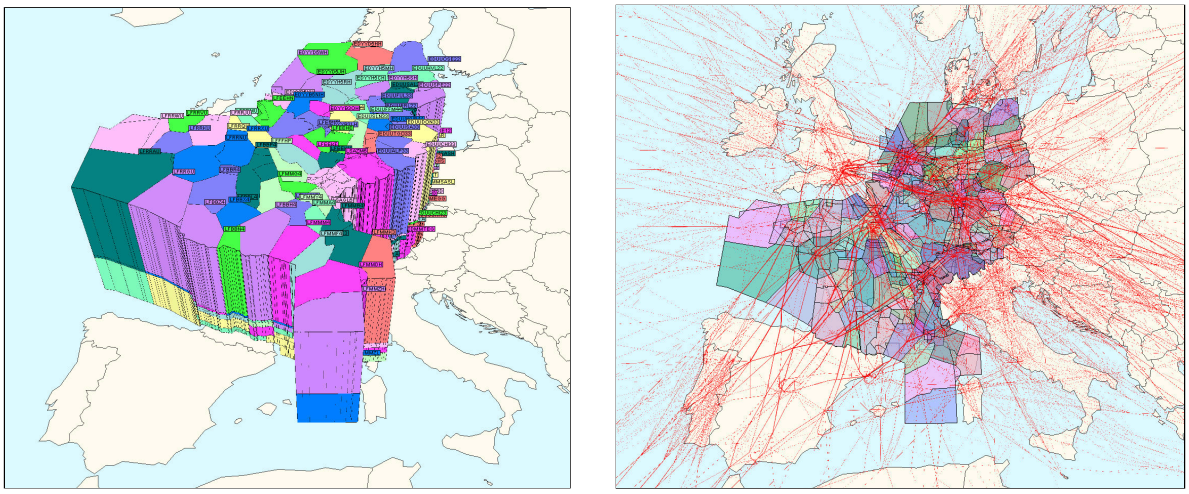
This section presents the numerical experiments performed under the C-ATFM framework. A real-world case study has been considered, focused on the Functional Airspace Block Europe Central (FABEC) airspace. FABEC covers the lower and upper airspace of six States (Belgium, France, Germany, Luxembourg, the Netherlands and Switzerland) and is one of the busiest and most complex in the world, handling about 55% of European air traffic.

A. Experimental scenario setup

The source data used in the experiments, corresponding to a typical day in February 2017, have been collected from the Demand Data Repository v2 (DDR2) published by EUROCONTROL. These mainly include the information of 337 elementary sectors within the FABEC (see Fig. 5a), and the corresponding traffic therein which accounted for 13,098 flights on that day (the total number is 26,840 for the whole European area), as shown in Fig. 5b.

Firstly, the initial 13,098 aircraft trajectories are generated in Module I, followed by the hotspot detection of Module II where there are 392 different operating sectors (which are functioned or collapsed, per 20 min, by the 337 elementary sectors) throughout the day. Among the set of operating sectors opened each 20 min period of time, i.e., 6,993 in total, we identify 107 of them having notably higher traffic demand than 1.5x declared capacity (namely those having severe imbalances), which in turn are regarded as the time-varing hotspots. Next, in Module III, the captured 1,920 flights (that are initially scheduled to fly across any of the hotspots) are allowed to submit their updated trajectories as the alternative options, considering the hotspot information. An additional set of 3,107 trajectories with regard to 1,764 different flights are submitted, resulting to 16,205 trajectories scheduled for 13,098 flights. Note that these simulated trajectories are produced using the same way as we generate the initial trajectories, meeting the hotspot-avoidance requirements.

In addition, some key assumptions have been made in this case study: (a) the unit time is set to 1 min; (b) the time scale for balancing demand and capacity is 20 min; (c) the cost for ground delay is considered as 81 Euro/min, airborne



(a) FABEC elementary sectors (3D vision)

(b) FABEC handled traffic (2D vision)

Fig. 5 Experimental scenario of the FABEC airspace (source: EUROCONTROL NEST).

delay 90 Euro/min, delay recovery -5 Euro/min, fuel price 0.8 Euro/kg, and route charges are based on real unit costs for different states with its weighting cost set to 1; and (d) the upper bound for performing airborne delay is 20% of the segment flight time, and for delay recovery this bound is set to 10%. Sensitivity experimentation has been performed in [16] with regard to some of the most important parameters to assess their effects.

Taking all above considerations into account, the original DCB optimiser of Module IV is executed to attempt the optimal solution (as done previously in [16] using a much smaller illustrative example). We will use the original DCB optimiser to tackle the FABEC case study, as the benchmark for this paper.

A set of Amazon Elastic Compute Cloud (EC2) instances are set up to provide a virtual computing environment, where various EC2 instance types (as summarised in Table 1) are used. Specifically, the "r4" family shown in the table is optimised for memory-intensive applications, while the "m5" is the latest generation of general purpose instances, both powered by the Intel Xeon series CPUs. EC2 instances support multi-threading, which enables multiple threads to run concurrently on a single CPU core. Each thread is represented as a virtual CPU (vCPU). All instances in this study are with an Ubuntu Server 18.04 OS. GAMS v25.03 is used as the modelling tool, and GUROBI v7.52 as the solver.

Table 1 Amazon EC2 instance types used in this study.

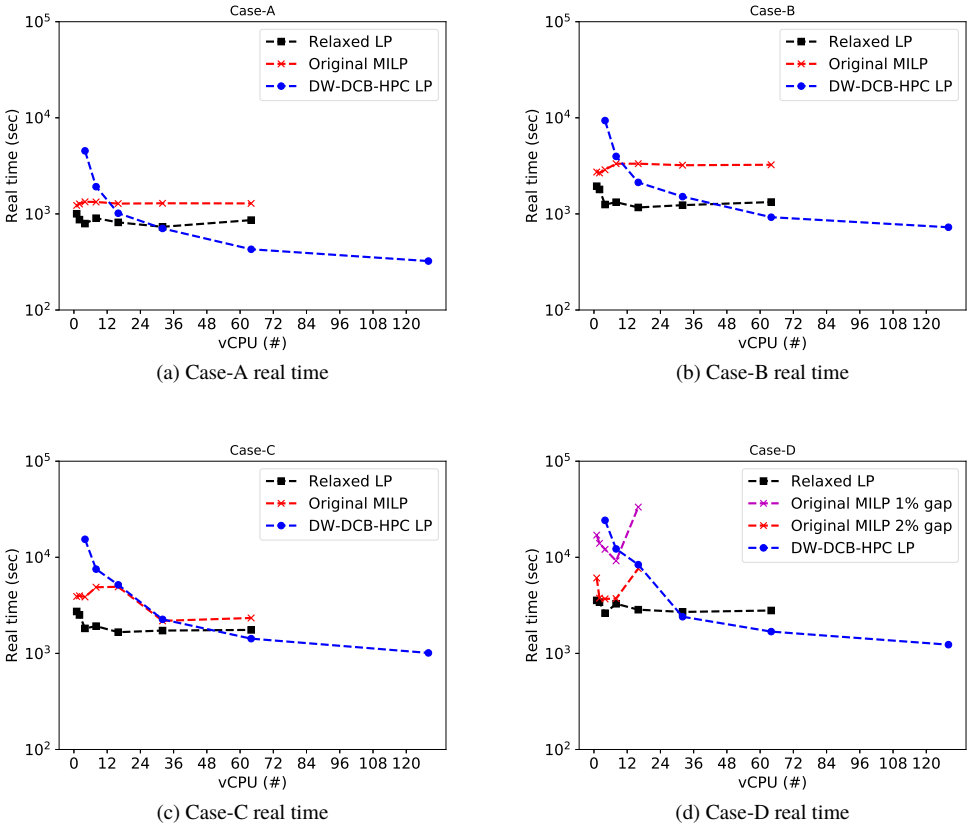
Instance	CPU model	vCPU (#)	Mem (GiB)	Networking (Gbps)	Purposes
r4.xlarge	E5-2686 2.3 GHz	4	30.5	Up to 10	Relaxed LP & Original MILP
r4.2xlarge	E5-2686 2.3 GHz	8	61	Up to 10	Relaxed LP & Original MILP
r4.4xlarge	E5-2686 2.3 GHz	16	122	Up to 10	Relaxed LP & Original MILP
r4.8xlarge	E5-2686 2.3 GHz	32	244	10	Relaxed LP & Original MILP
r4.16xlarge	E5-2686 2.3 GHz	64	488	25	Relaxed LP & Original MILP
m5.xlarge	Platinum 8175 2.5 GHz	4	16	Up to 10	DW-DCB-HPC Relaxed LP
m5.2xlarge	Platinum 8175 2.5 GHz	8	32	Up to 10	DW-DCB-HPC Relaxed LP

B. Results

The principal metrics for this study concern runtime and solution quality. In order to have a set of comparable results, four different case studies are designed based on the same experimental scenario introduced in Sec. V.A. The case studies are generated by varying two key parameters: sector capacity (C_f^τ) and feasible time window ($\mathcal{T}_k^j$). The different parameter values are given in Table 2, as well as the model dimensions with respect to the original problem. In terms of the program runtime, we consider three default measures used by the operating system, including *real* time (wall clock time), *user* time and *sys* time that both reflect the CPU-seconds.

Table 2 Case studies for the original MILP problem.

Case	Capacity variant (C_f^τ)	Time window ($\mathcal{T}_k^j$)	Variables	Equations	Non-zeros
Case-A	120%	20 min	5,099,570	11,593,326	27,957,776
Case-B	100%	30 min	7,410,190	17,052,320	40,964,362
Case-C	90%	35 min	8,565,500	19,781,818	47,467,626
Case-D	80%	40 min	9,720,810	22,511,312	53,969,964



(a) Case-A real time

(b) Case-B real time

(c) Case-C real time

(d) Case-D real time

Fig. 6 Real time used for solving each problem with different numbers of vCPU.

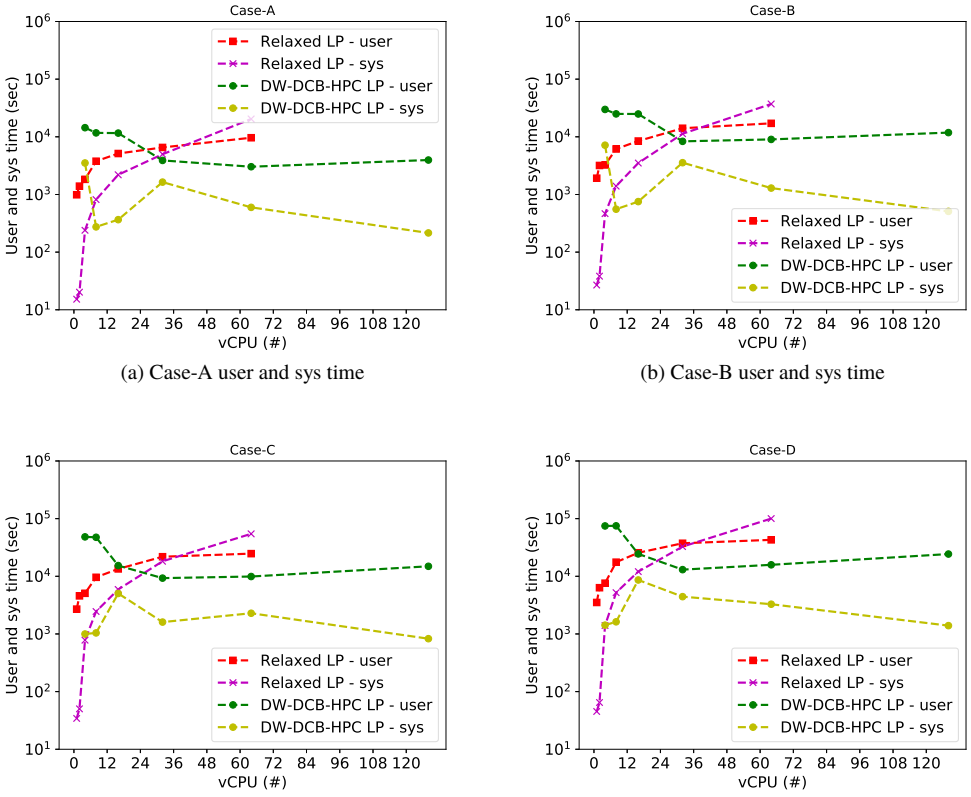
1. Runtime comparison

The large-scale models in Table 2 are first solved using directly the GAMS/GUROBI solver for both the relaxed LP and original MILP problems. Given the size of the models, we choose the "r4" family instances (with up to 64 vCPUs as shown in Table 1). The multi-threading concurrent solves consume a significant amount of extra memory, and thus the more processors are engaged in a single instance the more memory is needed. Next, the DW-DCB-HPC program is installed within the EC2 virtual machine cluster, and then executed for the different case studies. The decomposed models require significantly less memory, and thus we can choose a more balanced "m5.xlarge" computing instance which has 4 vCPUs with 2.5 GHz frequency and 16 GiB memory. A set of "m5.xlarge" instances (nodes) are connected to make up a large amount of vCPUs (up to 128 in this study) as a whole.

The real time is presented in Fig. 6. As the problem becomes more complicated over different cases, the runtime increases dramatically. Similar characteristics can be observed (except for Case-D). The runtime for solving the relaxed LP problem remains stable. The best choice for this problem seems at 4 vCPUs when applying the multi-threading concurrent solves, while no additional gains in time saving can be obtained regardless of having more cores engaged. For MILP solutions, the runtime also remains constant when the problem is easy to solve (Case-A and Case-B), but it turns unpredictable over the number of cores when solving challenging problems (Case-C and Case-D). For Case-D, it returns no MILP solution within a reasonable time when using more than 16 vCPUs. Fig. 6d also shows that significant more time is needed to improve the solution from 2% to 1% optimal gap.

The trend for DW-DCB-HPC solutions, as shown for all the cases in Fig. 6, is clear and also predictable, namely the more cores are used the faster the algorithm can be executed. This means that the parallelism is effective. When having a small number of cores, using the direct concurrent solves is much faster, but with approximately 32 vCPUs, the DW-DCB-HPC solution can overtake the concurrent solves. Moreover, such advantage tends to be larger with more cores involved. However, the marginal gains in time saving against the number of cores are gradually declining, which is mainly due to the extra computing sources needed for conducting data exchange through the network.

The results of user time and the sys time are shown in Fig. 7, where the "Relaxed LP" and the "DW-DCB-HPC LP" terms correspond to the black and blue curves in Fig. 6 respectively. We can notice that if using the concurrent solves, both times grow constantly along with the increase of core numbers. Taking into account that the real time does not change significantly, this means that the solution speed is constrained by the fastest solving method while some additional computing resources (used for running other methods) are not effective in the sense of parallelism. On the other side, running the DW-DCB-HPC algorithm consumes only a small amount of sys time (which even decreases with more cores involved), meaning that most resources are used for executing the computing tasks. The user time declines fast at the beginning (i.e., the same trending as per the real time), but it increases when having a large amount of cores. This difference is roughly similar to the change in marginal reduction of the real time (see Fig. 6).



(a) Case-A user and sys time

(b) Case-B user and sys time

(c) Case-C user and sys time

(d) Case-D user and sys time

Fig. 7 Other runtime measures for solving the LP problem with different numbers of vCPU.

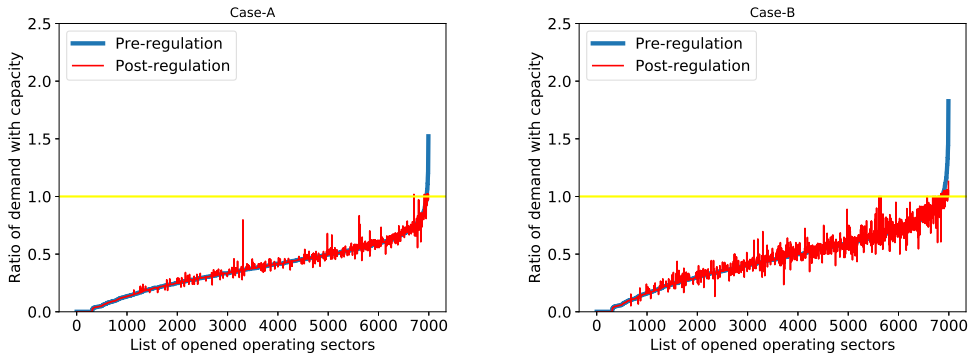
2. Solution quality assessment

It has been clarified that in practise, due to the uncertainty level of pre-tactical ATFM operations, a small number of capacity overloads are usually allowed. In this section, we assess how many capacity constraints are initially broken (i.e. pre-solution) and how many capacity overloads still exist after the DCB (i.e. post-solution). Recall that all the flight trajectory and time relevant constraints are always kept satisfied.

Table 3 Number of broken constraints and maximum capacity overload per sector.

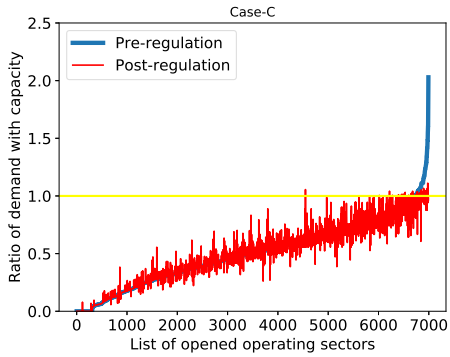
Solution	**Pre-solution**		**Post-solution**	
Case	Broken constraints (#)	Max overload (a/c)	Broken constraints (#)	Max overload (a/c)
Case-A	36	10	25	1
Case-B	107	13	39	2
Case-C	256	15	101	2
Case-D	514	17	259	2

Fig. 8 shows the ratios of demand to capacity, sorted based on the initial demand, with regard to running the DW-DCB-HPC program. In terms of Case-A and Case-B, there appear only a few capacity overloads, most of which lie at the sectors that are initially congested. For Case-C and Case-D, more broken constraints can be observed.

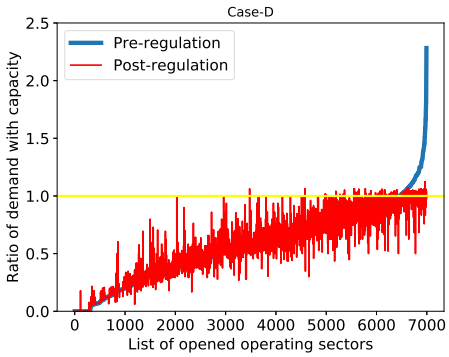


(a) Case-A DW-DCB-HPC LP

(b) Case-B DW-DCB-HPC LP



(c) Case-C DW-DCB-HPC LP



(d) Case-D DW-DCB-HPC LP

Fig. 8 Ratios of demand to capacity derived from running the DW-DCB-HPC program.

Nevertheless, most of the overloads are less than 5% beyond the declared capacity.

Specifically, the number of broken constraints and the associated maximal capacity overload are given in Table 3. It is guaranteed that no constraint violation will exist when attempting to solve the problem with the original DCB optimiser. After executing the DW-DCB-HPC program, around half of the broken constraints are mitigated. The maximal overload is significantly reduced from more than 10 aircraft to around 1-2 per sector. This suggests that the program is effective in balancing demand with capacity, only yielding a very small overload. Given the large uncertainty at the pre-tactical phase where the C-ATFM framework is aimed, such overload level could be fully tolerable.

VI. Conclusions

In this paper, we applied DW decomposition method and multi-level parallelism to the DCB optimiser considered in a previous C-ATFM framework [16], aiming at improving the TRL of the original works. The main improvement is in the computational tractability for this framework when tackling large-scale instances. A real-world case study (24 hours traffic crossing the FABEC area) suggests that, using the developed program, the computational performance can overtake running directly a state-of-the-art LP solver. The quality of the integral solution also shows that only a few

constraints broken in entry count capacity with tolerable overloads.

Nevertheless, there are still some operational barriers to be addressed. The major issues include the uncertainty and fairness concerns. Both require an extension or even remodelling with regard to the core DCB problem. For instance, the model may be revised into a multi-stage dynamic stochastic model. This will additionally impose significant complexity on the computing tasks. The parallelism paves the way towards quickly solving this complicated problem. In addition, the GAMS suite is relatively heavy to be executed in parallel. The performance could be further improved with a lighter modelling tool and transferring most computing tasks from CPU to GPU, which could also help overcome the network barriers. Also, the reactionary delay has not been considered in this paper, as modelling the turn around processes (in addition to the ATFM process) might be complex, which we believe deserves a separate work in the future.

Acknowledgement

The work presented in this paper was partially funded by the SESAR Joint Undertaking under grant agreement No. 699338, as part of the European Unions Horizon 2020 research and innovation programme: APACHE project (Assessment of Performance in current ATM operations and of new Concepts of operations for its Holistic Enhancement). The opinions expressed herein reflect the authors view only. Under no circumstances shall the SESAR Joint Undertaking be responsible for any use that may be made of the information contained herein. We would like to thank Prof. Gokhan Inalhan from Cranfield University and Dr. Christian Verdonk from CRIDA A.I.E. for their insightful review comments.

References

[1] ICAO, *Manual on Collaborative Decision-Making (CDM). Doc 9971 AN/485*, International Civil Aviation Organization, Montréal, Quebec, Canada, 2012.

[2] Bertsimas, D., and Gupta, S., "Fairness and collaboration in network air traffic flow management: an optimization approach," *Transportation Science*, Vol. 50, No. 1, 2015, pp. 57–76. doi:10.1287/trsc.2014.0567.

[3] Bertsimas, D., and Patterson, S. S., "The air traffic flow management problem with enroute capacities," *Operations research*, Vol. 46, No. 3, 1998, pp. 406–422. doi:10.1287/opre.46.3.406.

[4] Rios, J., and Ross, K., "Solving high fidelity, large-scale traffic flow management problems in reduced time," *The 26th Congress of ICAS and 8th AIAA ATIO*, 2008, p. 8910. doi:10.2514/6.2008-8910.

[5] Rios, J., and Ross, K., "Massively parallel Dantzig-Wolfe decomposition applied to traffic flow scheduling," *Journal of Aerospace Computing, Information, and Communication*, Vol. 7, No. 1, 2010, pp. 32–45. doi:10.2514/1.45606.

[6] Tandale, M. D., Wiraatmadja, S., Vaddi, V. V., and Rios, J. L., "Massively Parallel Optimal Solution to the Nationwide Traffic Flow Management Problem," *2013 Aviation Technology, Integration, and Operations Conference*, Los Angeles, CA, US, 2013, p. 4349. doi:10.2514/6.2013-4349.

[7] Balakrishnan, H., and Chandran, B. G., "Optimal Large-Scale Air Traffic Flow Management," Tech. rep., Massachusetts Institute of Technology, 2014.

[8] Sun, D., Clinet, A., and Bayen, A. M., "A dual decomposition method for sector capacity constrained traffic flow optimization," *Transportation Research Part B: Methodological*, Vol. 45, No. 6, 2011, pp. 880–902. doi:10.1016/j.trb.2011.03.004.

[9] Wei, P., Cao, Y., and Sun, D., "Total unimodularity and decomposition method for large-scale air traffic cell transmission model," *Transportation Research Part B: Methodological*, Vol. 53, 2013, pp. 1–16. doi:10.1016/j.trb.2013.03.004.

[10] Cao, Y., and Sun, D., "A Parallel Computing Framework for Large-Scale Air Traffic Flow Optimization," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 13, No. 4, 2012, pp. 1855–1864. doi:10.1109/TITS.2012.2205145.

[11] Cao, Y., and Sun, D., "Migrating large-scale air traffic modeling to the cloud," *Journal of Aerospace Information Systems*, Vol. 12, No. 2, 2015, pp. 257–266. doi:10.2514/1.I010150.

[12] Chen, J., Cao, Y., and Sun, D., "Modeling, optimization, and operation of large-scale air traffic flow management on spark," *Journal of Aerospace Information Systems*, Vol. 14, No. 9, 2017, pp. 504–516. doi:10.2514/1.I010533.

[13] FAA, "Collaborative Trajectory Options Program (CTOP): Document Information." Tech. Rep. AC 90-115, Federal Aviation Administration, 2014.

[14] Miller, M. E., and Hall, W. D., "Collaborative Trajectory Option Program demonstration," *Proceedings of the 34th IEEE/AIAA Digital Avionics Systems Conference (DASC)*, IEEE, Prague, Czech Republic, 2015, pp. 1C1–8. doi:10.1109/DASC.2015.7311339.

[15] Rodionova, O., Arneson, H., Sridhar, B., and Evans, A., "Efficient trajectory options allocation for the collaborative trajectory options program," *Proceedings of the 36th IEEE/AIAA Digital Avionics Systems Conference (DASC)*, St. Petersburg, FL, US, 2017, pp. 1–10. doi:10.1109/DASC.2017.8101997.

[16] Xu, Y., Dalmau, R., Melgosa, M., de Montlaur, A., and Prats, X., "A Framework for Collaborative Air Traffic Flow Management Minimizing Costs for Airspace Users: Enabling Trajectory Options and Flexible Pre-tactical Delay Management," *Transportation Research Part B: Methodological*, Vol. 134, 2020, pp. 229–255. doi:10.1016/j.trb.2020.02.012.

[17] Xu, Y., Prats, X., and Delahaye, D., "Synchronised Demand and Capacity Balancing in Collaborative Air Traffic Flow Management," *Transportation Research Part C: Emerging Technologies*, Vol. 114, 2020, pp. 359–376. doi:10.1016/j.trc.2020.02.007.

[18] Dantzig, G. B., and Wolfe, P., "Decomposition principle for linear programs," *Operations research*, Vol. 8, No. 1, 1960, pp. 101–111. doi:10.1287/opre.8.1.101.

[19] Wolsey, L. A., and Nemhauser, G. L., *Integer and combinatorial optimization*, Vol. 55, John Wiley & Sons, 1999.

[20] Vanderbeck, F., and Savelsbergh, M. W., "A generic view of Dantzig–Wolfe decomposition in mixed integer programming," *Operations Research Letters*, Vol. 34, No. 3, 2006, pp. 296–306. doi:10.1016/j.orl.2005.05.009.

[21] EUROCONTROL, "Study of the impact of ATFM uncertainty and smoothing performance on declared capacity," Tech. Rep. EEC Note No. 21/2006, EUROCONTROL Experimental Centre, 2006.

[22] Dalcín, L., Paz, R., Storti, M., and D'Elía, J., "MPI for Python: Performance improvements and MPI-2 extensions," *Journal of Parallel and Distributed Computing*, Vol. 68, No. 5, 2008, pp. 655–662. doi:10.1016/j.jpdc.2007.09.005.

[23] GAMS Development Corporation, "General Algebraic Modeling System (GAMS) Release 24.2.1," Washington, DC, USA, 2013. URL `http://www.gams.com/`.

[24] Gurobi Optimization, L., "Gurobi Optimizer Reference Manual," , 2019. URL `http://www.gurobi.com`.

Nomenclature

$f \in \mathcal{F}$	=	set of flights
$k \in \mathcal{K}$	=	set of trajectories
$j \in \mathcal{J}$	=	set of control points
$t \in \mathcal{T}$	=	set of time moments
$l \in \mathcal{L}$	=	set of operating sectors
$\tau \in \mathbb{T}$	=	set of time periods
$\mathcal{K}_f$	=	subset of trajectory options of flight f
$\mathcal{T}_k^j$	=	subset of feasible time window for trajectory k at position j
$\mathcal{L}_\tau$	=	subset of operating sectors that are open in time period τ
$\mathcal{J}_k$	=	subset of control points that trajectory k traverses
$\mathcal{J}_k^{(m)}$	=	$\begin{cases} \text{departure airport,} & \text{if } m = 1 \\ \text{arrival airport,} & \text{if } m = n \\ \text{intermediate designed positions,} & \text{if } 1 < m < n \end{cases}$
$\mathcal{T}(\tau)$	=	subset of time moments subject to time period τ
r_k^j	=	initially scheduled time of trajectory k at position j
$\underline{\mathcal{T}}_k^j$	=	lower bound of the feasible time window $\mathcal{T}_k^j$
$\overline{\mathcal{T}}_k^j$	=	upper bound of the feasible time window $\mathcal{T}_k^j$
$\underline{\mathcal{K}}_f$	=	first trajectory option of flight f
$\overline{\mathcal{K}}_f$	=	last trajectory option of flight f
$\underline{T}_k^{jj'}$	=	time bound of delay recovery within flight segment (j, j')
$\overline{T}_k^{jj'}$	=	time bound of air holding within flight segment (j, j')
$\mathcal{J}_{k,l}^\tau$	=	the first elementary sector entry for trajectory k within operating sector l in time period τ

C_l^τ	=	capacity of operating sector l during time period τ
α, β	=	weighting cost for fuel consumption and route charges respectively
γ^m	=	weighting cost for delays, where ground delay $m = g$, air delay $m = a$ and delay recovery $m = d$
ϵ	=	fairness factor for equivalent delay assignment
$i \in \mathcal{I}$	=	set of tentative proposals
$\Theta \subset \mathcal{I}$	=	subset of valid proposals
Θ_f	=	subset of valid proposals of f^{th} subproblem
$\pi_1^{\tau,l}$	=	dual of coupling constraints in line with sector capacity C_l^τ
$\pi_2^{f,x}, \pi_2^{f,y}, \pi_2^{f,z}$	=	dual of f^{th} subproblem's convexity constraint for decision variables corresponding to x, y, z
$\lambda_{f,i}^x, \lambda_{f,i}^y, \lambda_{f,i}^z$	=	weights of extreme points for decision variables corresponding to x, y, z

2021-05-21

Fast-time demand-capacity balancing optimizer for collaborative air traffic flow managem

Xu, Yan

AIAA

Xu Y, Camargo L, Prats X. (2021) Fast-time demand-capacity balancing optimizer for collaborative air traffic flow management. Journal of Aerospace Information Systems, Volume 18, Issue 8, August 2021, pp. 583-

https://doi.org/10.2514/1.I010948